Postel

Network Working Group
Request for Comments: 1146
Obsoletes: RFC 1145

J. Zweig
UIUC
C. Partridge
BBN
March 1990

TCP Alternate Checksum Options

Status of This Memo

This memo suggests a pair of TCP options to allow use of alternate data checksum algorithms in the TCP header. The use of these options is experimental, and not recommended for production use.

Note: This RFC corrects errors introduced in the editing process in RFC 1145.

Distribution of this memo is unlimited.

Introduction

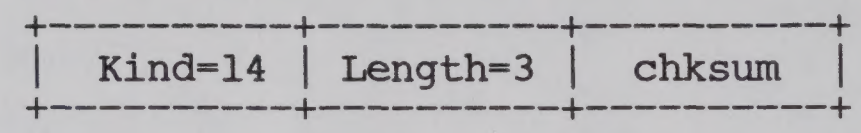
Some members of the networking community have expressed interest in using checksum-algorithms with different error detection and correction properties than the standard TCP checksum. The option described in this memo provides a mechanism to negotiate the use of an alternate checksum at connection-establishment time, as well as a mechanism to carry additional checksum information for algorithms that utilize checksums that are longer than 16 bits.

Definition of the Options

The TCP Alternate Checksum Request Option may be sent in a SYN segment by a TCP to indicate that the TCP is prepared to both generate and receive checksums based on an alternate algorithm. During communication, the alternate checksum replaces the regular TCP checksum in the checksum field of the TCP header. Should the alternate checksum require more than 2 octets to transmit, the checksum may either be moved into a TCP Alternate Checksum Data Option and the checksum field of the TCP header be sent as 0, or the data may be split between the header field and the option. Alternate checksums are computed over the same data as the regular TCP checksum (see TCP Alternate Checksum Data Option discussion below).

TCP Alternate Checksum Request Option

The format of the TCP Alternate Checksum Request Option is:



Here chksum is a number identifying the type of checksum to be used.

The currently defined values of chksum are:

0 -- TCP checksum
1 -- 8-bit Fletcher's algorithm (see Appendix I)
2 -- 16-bit Fletcher's algorithm (see Appendix II)

Note that the 8-bit Fletcher algorithm gives a 16-bit checksum and the 16-bit algorithm gives a 32-bit checksum.

Alternate checksum negotiation proceeds as follows:

A SYN segment used to originate a connection may contain the Alternate Checksum Request Option, which specifies an alternate checksum-calculation algorithm to be used for the connection. The acknowledging SYN-ACK segment may also carry the option.

If both SYN segments carry the Alternate Checksum Request option, and both specify the same algorithm, that algorithm must be used for the remainder of the connection. Otherwise, the standard TCP checksum algorithm must be used for the entire connection. Thus, for example, if one TCP specifies type 1 checksums, and the other specifies type 2 checksums, then they will use type 0 (the regular TCP checksum). Note that in practice, one TCP will typically be responding to the other's SYN, and thus either accepting or rejecting the proposed alternate checksum algorithm.

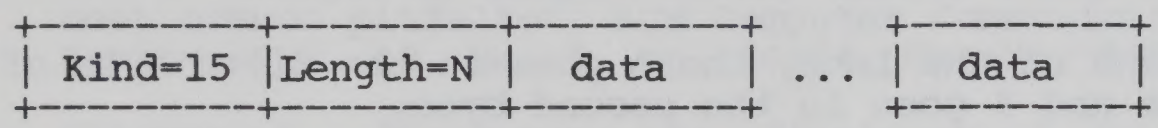
Any segment with the SYN bit set must always use the standard TCP checksum algorithm. Thus the SYN segment will always be understood by the receiving TCP. The alternate checksum must not be used until the first non-SYN segment. In addition, because RST segments may also be received or sent without complete state information, any segment with the RST bit set must use the standard TCP checksum.

The option may not be sent in any segment that does not have the SYN bit set.

An implementation of TCP which does not support the option should silently ignore it (as RFC 1122 requires). Ignoring the option will force any TCP attempting to use an alternate checksum to use the standard TCP checksum algorithm, thus ensuring interoperability.

TCP Alternate Checksum Data Option

The format of the TCP Alternate Checksum Data Option is:



This field is used only when the alternate checksum that is negotiated is longer than 16 bits. These checksums will not fit in the checksum field of the TCP header and thus at least part of them must be put in an option. Whether the checksum is split between the checksum field in the TCP header and the option or the entire checksum is placed in the option is determined on a checksum by checksum basis.

The length of this option will depend on the choice of alternate checksum algorithm for this connection.

While computing the alternate checksum, the TCP checksum field and the data portion TCP Alternate Checksum Data Option are replaced with zeros.

An otherwise acceptable segment carrying this option on a connection using a 16-bit checksum algorithm, or carrying this option with an inappropriate number of data octets for the chosen alternate checksum algorithm is in error and must be discarded; a RST-segment must be generated, and the connection aborted.

Note the requirement above that RST and SYN segments must always use the standard TCP checksum.

APPENDIX I: The 8-bit Fletcher Checksum Algorithm

The 8-bit Fletcher Checksum Algorithm is calculated over a sequence of data octets (call them D[1] through D[N]) by maintaining 2 unsigned 1's-complement 8-bit accumulators A and B whose contents are initially zero, and performing the following loop where i ranges from 1 to N:

```
A := A + D[i]
B := B + A
```

It can be shown that at the end of the loop A will contain the 8-bit 1's complement sum of all octets in the datagram, and that B will contain (N)D[1] + (N-1)D[2] + ... + D[N].

The octets covered by this algorithm should be the same as those over

which the standard TCP checksum calculation is performed, with the pseudoheader being D[1] through D[12] and the TCP header beginning at D[13]. Note that, for purposes of the checksum computation, the checksum field itself must be equal to zero.

At the end of the loop, the A goes in the first byte of the TCP checksum and B goes in the second byte.

Note that, unlike the OSI version of the Fletcher checksum, this checksum does not adjust the check bytes so that the receiver checksum is 0.

There are a number of much faster algorithms for calculating the two octets of the 8-bit Fletcher checksum. For more information see [Sklower89], [Nakassis88] and [Fletcher82]. Naturally, any computation which computes the same number as would be calculated by the loop above may be used to calculate the checksum. One advantage of the Fletcher algorithms over the standard TCP checksum algorithm is the ability to detect the transposition of octets/words of any size within a datagram.

APPENDIX II: The 16-bit Fletcher Checksum Algorithm

The 16-bit Fletcher Checksum algorithm proceeds in precisely the same manner as the 8-bit checksum algorithm,, except that A, B and the D[i] are 16-bit quantities. It is necessary (as it is with the standard TCP checksum algorithm) to pad a datagram containing an odd number of octets with a zero octet.

Result A should be placed in the TCP header checksum field and Result B should appear in an TCP Alternate Checksum Data option. This option must be present in every TCP header. The two bytes reserved for B should be set to zero during the calculation of the checksum.

The checksum field of the TCP header shall contain the contents of A at the end of the loop. The TCP Alternate Checksum Data option must be present and contain the contents of B at the end of the loop.

BIBLIOGRAPHY:

[BrBoPa89] Braden, R., Borman, D., and C. Partridge, "Computing the Internet Checksum", ACM Computer Communication Review, Vol. 19, No. 2, pp. 86-101, April 1989. [Note that this includes Plummer, W. "IEN-45: TCP Checksum Function Design" (1978) as an appendix.]

[Fletcher82] Fletcher, J., "An Arithmetic Checksum for Serial Transmissions", IEEE Transactions on Communication,

Vol. COM-30, No. 1, pp. 247-252, January 1982.

[Nakassis88] Nakassis, T., "Fletcher's Error Detection Algorithm: How to implement it efficiently and how to avoid the most common pitfalls", ACM Computer Communication Review, Vol. 18, No. 5, pp. 86-94, October 1988.

[Sklower89] Sklower, K., "Improving the Efficiency of the OSI Checksum Calculation", ACM Computer Communication Review, Vol. 19, No. 5, pp. 32-43, October 1989.

Security Considerations

Security issues are not addressed in this memo.

Authors' Addresses

Johnny Zweig
Digital Computer Lab
University of Illinois (UIUC)
1304 West Springfield Avenue
CAMPUS MC 258
Urbana, IL 61801

Phone: (217) 333-7937

EMail: zweig@CS.UIUC.EDU

Craig Partridge
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02138

Phone: (617) 873-2459

EMail: craig@BBN.COM